PRÉSENTATION MEMORY GAME

COMMENT JE CONSTRUIS MON CRUD

CRUD = create | read | update | delete = acronyme mnémotechnique pour les façons dont on peut fonctionner sur des données stockées, ici 4 fonctions de bases du stockage persistant.

1. Définir les fondations de mon projet
    a. Langage de programmation et framework ? j'utilise **JavaScript avec Node.js/Express**
    b. BDD ? j'utilise **SQL | MySQL**
    c. Structure de données ? Quels sont les modèles de données principaux à gérer dans mon application? Je dois décrire les principales entités et leurs attributs. Je ne sais pas répondre à ça, aide- moi.
    d. Architecture ? j'utilise une architecture **MVC**, comment j'organise mes fichiers dans mon projet ?
    e. Fonctionnalités spécifiques ? Qu'est ce que je veux implémenter en plus des opérations CRUD de base (création, lecture, mise à jour, suppression ?).
    f. Interface utilisateur ? Mes préférences pour la création ? J'utilise un framework frontend React.

2. Définir les Modèles de Données
    a. Draw.io
    b. StarUML

3. Créer ma BDD
    a. lancer wamp
    b. go to phpMyAdmin

4. Mettre en place mon architecture MVC
   j'installe les dépendances principales, React pour la création d'interfaces utilisateur, React DOM, React Router DOM pour gérer la navigation dans les applications React.

   CRUD suppléments :
    a. MAJ profil utilisateur
    b. MAJ scores de partie
    c. Suppression profil utilisateur

5. Fonctionnalités spécifiques

QUESTIONS À SE POSER POUR CONTINUER À CONSTRUIRE MON PROJET

1. Configuration du Serveur

Connexion à la base de données
Ai-je configuré la connexion à MySQL dans le fichier database.js dans config/ ?

Sécurité
Ai-je configuré des variables d'environnement sécurisées dans le fichier .env ?

Mise en place du serveur
Ai-je configuré le fichier server.js pour démarrer le serveur Express et importer les routes ?

2. Routes et Contrôleurs

Routes utilisateur
Ai-je défini les routes pour les opérations CRUD des utilisateurs dans router/userRoutes.js ?

Contrôleurs utilisateur
Ai-je implémenté les méthodes pour gérer les requêtes dans controllers/userController.js ?

Routes des parties (games)
Ai-je défini les routes pour les opérations CRUD des parties dans router/gameRoutes.js ?

Contrôleurs des parties
Ai-je implémenté les méthodes pour gérer les requêtes dans controllers/gameController.js ?

3. Modèles de Données

Modèle Utilisateur
Ai-je défini le modèle utilisateur dans models/user.js ?

Modèle Partie
Ai-je défini le modèle partie dans models/game.js ?

4. Middlewares

Authentification
Ai-je des middlewares pour gérer l'authentification des utilisateurs ? (e.g., middlewares/authMiddleware.js)

5. Interface Utilisateur

Structure du Projet React
Ai-je structuré mon projet React avec des composants et des pages dans le dossier client-side ?

Connexion Backend/Frontend
Ai-je mis en place les appels API depuis React pour interagir avec ton backend Express ?

6. Tests

Tests Unitaires
Ai-je mis en place des tests unitaires pour mes contrôleurs et modèles dans le dossier tests/ ?

Tests d'Intégration
Ai-je mis en place des tests d'intégration pour vérifier le bon fonctionnement de mon application complète ?

MÉTHODES D'IMPORTATIONS

Les différences entre les importations require et import sont principalement liées à la manière dont les modules sont chargés en JavaScript, et à la syntaxe moderne introduite par ECMAScript 6 (ES6). Voici quelques points clés :

Syntaxe :
require est la méthode de chargement de module de CommonJS, principalement utilisée dans Node.js.

import est la méthode de chargement de module introduite par ES6, qui est maintenant standardisée et largement utilisée dans les projets modernes de JavaScript.

Support :
require fonctionne dans Node.js nativement.

import nécessite souvent une transpilation via Babel ou d'autres outils pour fonctionner dans Node.js, sauf si vous utilisez des modules ES (fichiers avec l'extension .mjs ou en utilisant "type": "module" dans package.json).

Asynchronisme :

require est synchrone.

import est asynchrone et vous permet d'utiliser import() dynamiquement pour charger des modules.